VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Desková hra BattleTech v prostředí internetu

Desk game BattleTech over internet

2015

Petr Šlachta

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Šlachta**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Desková hra BatleTech v porstředí internetu
Desk Game BatleTech Over Internet

Zásady pro vypracování:

Cílem práce je vytvořit hru BatleTech v prostředí internetu v jazyce java, která umožní editovat herní plochu (´mapu) a sestavování vlastních bojových jednotek.
Hra bude umožňovat:
1. Editor herních plánů
2. Hra proti jednoduché umělé inteligenci.
3. Hru více hráčů přes internet.
4. Editor konfigurace bojových robotů.
5. Program umožní připojení do současně vyvíjeného portálů her v jazyce Java.

Práce bude obsahovat:
1. Implementaci hry BatleTech.
2. Programátorskou dokumentaci řešení s využitím diagramů jazyka UML.
3. Uživatelskou dokumentaci aplikace.

Seznam doporučené odborné literatury:

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
[2] DARWIN, Ian F. Java cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9. Dostupné z: http://it-ebooks.info/book/2249/
Dále podle pokynů vedoucího práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava*.

V Ostravě 7. května 2015 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Rád bych tímto poděkoval **Ing. Davidu Ježkovi, Ph.D.** za jeho pomoc, rady a konzultace během vytváření této bakalářské práce.

Také bych rád poděkoval Amitu Patelovi za jeho pomoc s řešením některých použitých algoritmů.

Abstrakt

Tato bakalářská práce popisuje metody a technologie, které jsem použil pro vývoj implementace deskové hry BattleTech v jazyce Java. Program umožňuje hrát hru lokálně na jednom počítači nebo prostřednictvím internetu. Dále program umožňuje hráči vytvoření vlastního herního plánu, neboli mapy, a také vytvoření vlastních bojových robotů, které pak může využít ve hře. Implementace také obsahuje jednoduchou umělou inteligenci.

Klíčová slova: Java, BattleTech, desková hra, online hra, hra více hráčů

Abstract

This bachelor thesis describes methods and technologies that I used to develop an implementation of a board game BattleTech in Java language. The program allows user to play localy on his computer or over the internet. It also allows user to create his own game plan and his own battle robots that can be used in the game. The implementation contains simple artificial intelligence.

Keywords: Java, BattleTech, board game, online game, multiplayer

Seznam použitých zkratek a symbolů

1k6	–	Jedna šestistěnná hrací kostka
2k6	–	Dvě šestistěnné hrací kostky
BP	–	Bod pohybu
hex	–	Šestiúhelník, který představuje jedno herní políčko
OBR	–	Obrněný Bitevní Robot
RDD	–	Rakety/Raketomet dlouhého doletu
RKD	–	Rakety/Raketomet krátkého doletu
UI	–	Umělá inteligence
VZZ	–	Výsledný zásah zbraní
warrior	–	Pilot, který řídí OBRa
XML	–	Extensible Markup Language/rozšiřitelný značkovací jazyk
ZZZ	–	Základ zásahu zbraní

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Obsahem tohoto dokumentu je popis mé implementace deskové hry BattleTech. Text je rozdělen na čtyři hlavní části. V první se zabývám popisem hry a nastíním i zlomek z rozsáhlého světa BattleTechu. Ve druhé části popisuji samotnou implementaci z programátorského hlediska a zabývám se některými svými řešeními různých situací. Třetí část obsahuje uživatelskou dokumentaci a popisuje herní prvky a ovládání hry z pohledu uživatele. V poslední části pak rozebírám editor herních plánů a také editor robotů, které umožňují uživateli vytvářet vlastní herní mapy a zkonstruovat si vlastního robota podle stejných pravidel, která byla použita u robotů již existujících ve hře.

Možnost vytvořit si vlastní hru mě velmi lákala, protože jsem předpokládal, že to bude velice zajímavá zkušenost. Doteď jsem se totiž s hrami setkával pouze z uživatelské strany a chtěl jsem si tedy vyzkoušet pohled na hru ze strany programátora, který ji vytváří. Hra BattleTech byla moje první volba hned z několika důvodů.
Hlavním důvodem byl zřejmě fakt, že už když jsem se s touto hrou poprvé setkal před několika lety, napadlo mě, že by bylo skvělé, kdyby tato hra existovala na počítači. Desková verze totiž může být poměrně zdlouhavá, zejména při hře s větším počtem OBRů. Například určení, jestli zbraň zasáhla cíl a následně kam ho zasáhla a jak silně, zahrnuje výpočet obtížnosti zásahu, několik hodů kostkami, několik hledání v tabulkách a záznam poškození do karty zasaženého OBRa. Tento proces se pak opakuje pro každou použitou zbraň a každého OBRa.
Dalším důvodem byla úžasná propracovanost herních mechanismů, což mě na této hře vždy velmi bavilo. Například konstrukce OBRů má svá naprosto přesná pravidla, podle kterých jsou všichni vytvořeni. Každá součást OBRa má svou váhu, která odpovídá její užitečnosti. Díky tomu může mít každý OBR své přednosti, ale z nich zároveň vždy vyplývají i slabiny, jako třeba že těžký obrněný OBR vybavený silnými zbraněmi je velmi nepohyblivý, čehož může menší rychlejší OBR využít.
Posledním důvodem zřejmě byla podle mě poměrně malá rozšířenost této hry u nás, díky čemuž mi přišlo zajímavé vytvořit verzi této hry hratelnou i online přes internet proti jiným hráčům.

Vzhledem k velké komplexnosti této hry se mi bohužel nepodařilo zpracovat veškeré možnosti, které desková verze nabízí. Verze hry, kterou jsem naimplementoval, je vlastně spíše úvod do BattleTechu. Tato varianta je určená pro začátečníky na seznámení se základními herními mechanismy a neobsahuje řadu pokročilých možností. Oproti běžné verzi tohoto „odlehčeného módu“ jsem navíc přidal i fázi přehřívání, která podle mě dělá hru mnohem zajímavější.

Hru jsem se snažil průběžně testovat a opravovat případné chyby, bohužel se mi však určitě nepodařilo odhalit všechny. Do jisté míry je to způsobeno i tím, že jako autor vnímám hru jinak a vím, jak které části programu používat, takže na některé skryté chyby vůbec nenarazím a zřejmě se tak odhalí až v případném zkušebním provozu.

2 O hře BattleTech

2.1 Popis hry

Hra BattleTech je sci-fi desková hra, která se odehrává ve 31. století. Lidstvo se rozšířilo na mnoho jiných planet a války jsou vedeny pomocí technologií této doby, což znamená využívání OBRů. Tyto velké stroje jsou ovládány warriory a bojují proti sobě pomocí silných zbraní. Svět BattleTechu je velmi rozsáhlý a obsahuje kromě deskových her i řadu knih. Má dokonce svou vlastní historii. Více lze nalézt na oficiálních stránkách BattleTechu [1].
Ve hře se vždy jedná o souboj dvou znepřátelených stran, které spolu bojují až do chvíle, kdy jedna ze stran ztratí všechny své OBRy.

Na začátku hry určí hráč, který hru vytváří, maximální celkovou váhu všech OBRů jedné strany. Každá strana si pak může libovolně vybrat jakoukoli kombinaci OBRů tak, aby součet jejich hmotností nepřesahoval určené maximum. Tuto kombinaci je dobré volit pečlivě, protože každý OBR má své výhody i nevýhody, takže nelze říci, že čím větší OBR je, tím je i lepší.
OBRy obou stran jsou pak náhodně rozmístěny na herním plánu s tím, že OBRy jedné strany jsou umístěny k levému okraji, zatímco OBRy druhé strany jsou umístěny k pravému okraji. Pak začne samotná hra.

2.2 Průběh hry

Hra se dělí na jednotlivá kola, která mají předem určený průběh. Každé herní kolo začíná určením iniciativy a po ní následují fáze pohyb, útok palbou, přehřívání a konec kola.

2.2.1 Iniciativa

Ve fázi iniciativa se určuje, která strana bude v tomto kole táhnout vždy jako první. Obě strany hodí 2k6 a strana, která hodí víc, získává v tomto kole iniciativu. Strana, která prohrála, bude na tahu vždy jako první, což umožní straně s iniciativou reagovat na tahy soupeře.

2.2.2 Pohyb

Ve fázi pohyb střídavě obě strany pohybují se svými OBRy. Jako první provede tah vždy strana, která prohrála iniciativu. Vždy vybere jednoho svého OBRa a provede s ním pohyb podle pravidel, případně o něm prohlásí, že se v tomto kole nebude hýbat. Následuje pohyb OBRa druhé strany. Takto se obě strany střídají, dokud nebylo pohnuto, případně prohlášeno, že se nebudou hýbat, se všemi OBRy na herní mapě. Pokud strana, která je na tahu, nemá už žádného OBRa, se kterým by mohla v tomto kole pohybovat, je na tahu znovu druhá strana.

2.2.3 Útok palbou

Ve fázi útok palbou obě strany střídavě nahlašují útoky palbou svých OBRů. Nejdříve začíná strana, která prohrála iniciativu, a určí, který její OBR bude střílet, na jaký cíl bude střílet a které zbraně k tomu použije.
Poté je na tahu druhá strana, která také určí jednoho svého OBRa a rozhodne, na jaký cíl a jakými zbraněmi bude střílet. Útoky jsou vyhodnoceny ihned, avšak poškození, které způsobí, začne platit až po skončení fáze Útok palbou, protože OBRy střílí všichni současně, nehledě na pořadí, v jakém byly jejich útoky nahlášeny a spočítány. Jakmile jsou tedy nahlášeny a spočítány všechny útoky, dojde k zaznamenání poškození a projeví se případné efekty, které toto poškození způsobí.

2.2.4 Přehřívání

Ve fázi přehřívání se zjišťuje, jestli se OBRům změnila vnitřní teplota a výsledná teplota se zaznamená do údajů o OBRovi. Jakékoli negativní účinky, které vzniknou v důsledku přehřátí OBRa, začnou působit až po skončení této fáze kola.

2.2.5 Konec kola

Ve fázi konec kola se dokončují činnosti, které vyplynou z předchozích událostí daného kola. Například dojde k odstranění zničených OBRů z hracího plánu. Po dokončení všech činností, které je potřeba vykonat před ukončením kola, se hra přesune do následujícího kola, které opět začíná fází iniciativa.

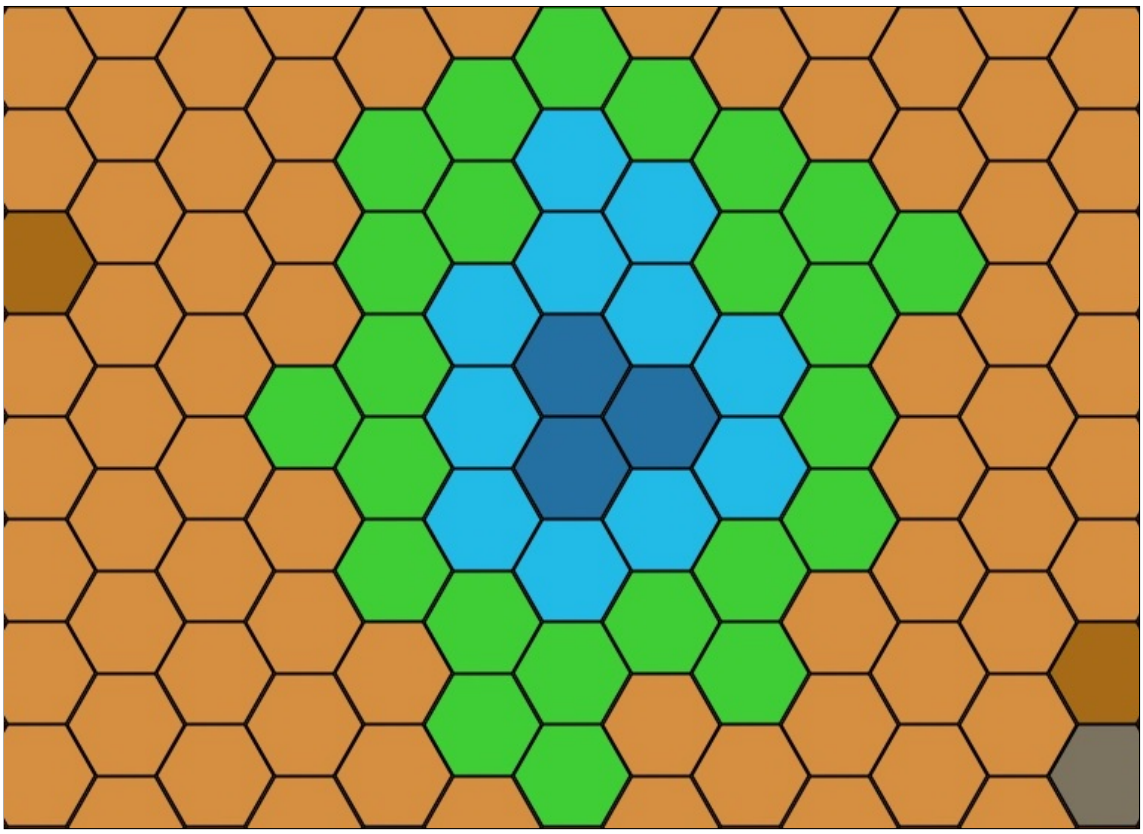
2.3 Herní mapa

Herní mapa nebo taky herní plán je hrací plocha, na které se odehrává souboj OBRů a představuje území, na kterém se odehrává bitva. Skládá se z šestiúhelníkových polí, kterým se říká hexy. Každý hex představuje prostor, který má napříč přibližně 30 metrů. Na každém hexu se může vždy nacházet pouze jeden OBR, který má hex pod svou taktickou kontrolou, i když sám o sobě samozřejmě nezabírá celý prostor hexu. Ukázku herní mapy je možné vidět na obrázku 1.

2.3.1 Terén

Každý hex může mít některý ze šesti základních typů terénu. Podle toho, jaký je terén hexu, se mění náročnost pohybu OBRa v tomto hexu. Terén může také výrazně ovlivnit viditelnost nebo přesnost palby ze střelných zbraní.

2.3.1.1 Lehký terén Základní typ terénu. Je snadné se po něm pohybovat a nijak nebrání ve výhledu ani neovlivňuje přesnost palby. Obsahuje pole a travnaté plochy.



Obrázek 1: Ukázka herní mapy a různých druhů terénu.

2.3.1.2 Těžký terén Tvoří ho kamenitá půda plná srázů, útesů a skalisek. Ve výhledu sice nebrání a ani nemá vliv na přesnost střelby, ovšem pohyb po tomto terénu je náročnější než pohyb po lehkém terénu.

2.3.1.3 Kopce Kopce nebo také hory jsou oblasti výrazně vyšší než jejich okolí. Vystoupat na vrchol takového kopce je obtížné a pohyb po tomto terénu je tedy náročnější, protože OBR musí překonat velký výškový rozdíl. Dostatečně vysoké kopce mohou úplně zakrýt výhled a tím i znemožnit palbu. Výška kopců může být různá a počítá se v úrovních. Čím vyšší úroveň hex má, tím vyšší je oproti běžnému terénu. Základní úroveň hexu je 0, každá vyšší úroveň je již považována za kopec.

2.3.1.4 Voda Podobně jako kopce má i voda své úrovně, kterým se říká hloubky. Čím větší je hloubka vody na hexu, tím níže pod základní úrovní se hex nachází. Voda větší hloubky než 0 má nepříznivý vliv na OBRův pohyb a voda větší hloubky než 2 stojícího robota zcela skryje, takže je nemožné ho vidět nebo na něj střílet.

2.3.1.5 Řídký les Povrch hexu je řídce porostlý stromy, které zhoršují viditelnost a přesnost střelby a pohyb lesem je náročnější než pohyb po lehkém terénu.

2.3.1.6 Hustý les Povrch hexu je pokryt hustými lesy, kterými je velmi obtížné projít a které téměř znemožňují viditelnost nebo střelbu přes tento hex.

2.4 OBR

OBR je stroj, který vládne bojištím 31. století. Slučuje velkou ničivou sílu se značnou obratností a rychlostí. Základ každého OBRa tvoří čtyři hlavní systémy - kostra, svaly, pancíř a výzbroj a zdroj energie. Kostra a svaly jsou z hlediska hry jako takové poměrně nepodstatné, ovšem pancíř a výzbroj a zdroj energie jsou velice důležitou součástí hry.

2.4.1 Pancíř a výzbroj

Každý OBR je chráněn dvěma různými druhy pancíře.
Vnější pancíř je velmi pevný, což mu umožňuje zastavit většinu běžných projektilů, a má také vysokou tepelnou vodivost, díky které je poměrně odolný vůči energetickým zbraním, jako jsou lasery nebo vrhače částic.
Vnitřní pancíř chrání vnitřní strukturu OBRa před úlomky zničeného vnějšího pancíře a před speciálními průraznými projektily.

Mezi nejběžnější zbraně patří zejména nejrůznější lasery a v menší míře vrhače částic. Tyto energetické zbraně mají velkou výhodu v tom, že nepotřebují munici a je proto možné je používat prakticky neomezeně. Tyto zbraně pak bývají doplněny různými typy raketometů či autokanónů.

2.4.2 Zdroj energie

Každý OBR neustále potřebuje obrovské množství energie pro pohon všech složitých systémů, pohyb a boj. K tomu se využívá fúzní vodíkový reaktor, který sice vyrábí velké množství energie, ale zároveň produkuje mnoho odpadního tepla, které může OBRa vážně poškodit. Aby se tomu zabránilo, je každý OBR vybaven chladiči, které toto teplo odvádějí.

2.5 Pohyb

Každý OBR má k dispozici až čtyři druhy pohybu, které může využít k pohybu po mapě - chůzi, běh, skok a stání. Chůze je základní druh pohybu, je pomalejší než běh, ale OBR neprodukuje téměř žádné přebytečné teplo. Běh je nejrychlejší druh pohybu, ale OBR při běhu produkuje více tepla než při chůzi. Skok je speciální druh pohybu, který vyžaduje zdvižné trysky, aby ho OBR mohl provést. Ne každý OBR však tyto trysky má, a proto ne každý OBR může skákat. Skok vyprodukuje nejvíce tepla ze všech druhů pohybu, ale může být nejefektivnějším způsobem překonání některých oblastí na mapě, protože

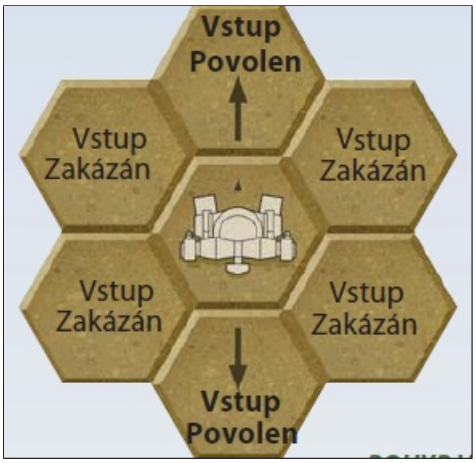
na něj nemají vliv různé typy terénu (viz 2.3.1). Pokud se OBR nechce pohybovat, může zůstat stát na místě. Stání neprodukuje vůbec žádné přebytečné teplo, je proto vhodné při snižování úrovně přehřátí OBRa.

2.5.1 Počítání pohybu

Rychlost OBRa se určuje pomocí BP. Jakýkoli pohyb pak vyžaduje určitý počet BP, aby ho OBR mohl provést. Kolik BP je zapotřebí záleží na typu terénu, na výškovém rozdílu mezi výchozím a cílovým hexem a také na typu pohybu.

2.5.2 Směr pohybu

OBR se může pohybovat z hexu na hex pouze směrem vpřed nebo vzad. Při pohybu vpřed se OBR přesune na hex, ke kterému stojí čelem, pokud na to má dostatek BP. Obdobně se může OBR posunout směrem vzad na hex, ke kterému je otočen zády. Na žádný jiný sousední hex OBR nemůže přímo přejít (viz obrázek 2). Aby mohl OBR jít na některý ze čtyř zbývajících sousedních hexů, musí se nejdříve otočit na svém hexu doleva nebo doprava. Každé otočení znamená pootočení OBRa čelem k jiné hraně, než ke které byl otočený předtím. OBR se může otočit vždy o jednu hranu doleva nebo doprava a každé takové otočení stojí jeden BP. Samozřejmě je možné tuto akci opakovat, dokud má OBR dostatek BP k jejímu provedení.



Obrázek 2: Ukázka možného pohybu OBRa.

2.6 Boj

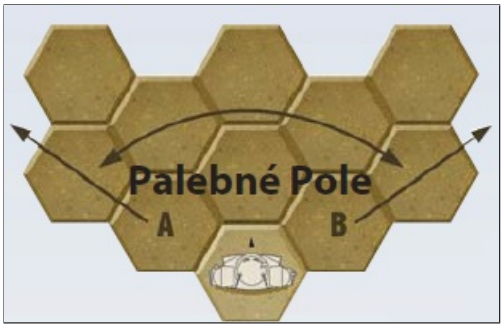
Boj spočívá v používání palných zbraní k poškození nepřátelských OBRů. Během fáze Útok palbou hráči střílí z každé zbraně všech svých OBRů, pokud chtějí. Z každé zbraně je však možné vystřelit pouze jedenkrát za kolo.

2.6.1 Linie výhledu

Aby mohl OBR střílet na jiného OBRa, musí ho vidět. Některé typy terénu mohou OBRovi překážet ve výhledu, čímž je útok těžší, nebo dokonce nemožný. Výhled OBRů může ovlivnit voda, hory nebo lesy. Linií výhledu je spojnice mezi středem hexu útočníka a středem hexu cíle, vedená vzdušnou čarou. Každý hex, kterým tato spojnice prochází, se počítá do linie výhledu, s výjimkou hexů, na kterých stojí útočník a jeho cíl.
Pokud OBR stojí ve vodě hloubky 2 a více, je celý ponořený pod vodu, což znamená, že má zcela zakrytou linii výhledu a tedy nemůže střílet. To samé však platí pro nepřátelské OBRy, kteří by na ponořeného OBRa chtěli vystřelit. Jelikož je OBR celý pod vodou, žádný jiný OBR ho nemůže vidět a tedy na něj nemůže střílet.
Pokud se v linii výhledu nachází lesy, výrazně to snižuje viditelnost. Jeden hex lesa sice ještě výhled nezakryje, ale tři a více hexů řídkého lesa už výhled zcela zakryjí. Pokud se na linii výhledu nachází alespoň jeden hex hustého lesa, každý další hex jakéhokoli lesa zakryje výhled úplně a OBR na svůj cíl nevidí. Lesy na hexech, na kterých stojí útočník a jeho cíl, samozřejmě neovlivní linii výhledu.
Pokud se mezi útočníkem a cílem nachází hex s úrovní alespoň o 2 úrovně vyšší než hexy, na kterých tito stojí, mají zakrytý výhled a nemohou na sebe střílet.

2.6.2 Palebné pole

Aby mohl OBR na svůj cíl střílet, musí se tento nepřátelský OBR nacházet v OBRově palebném poli. Palebné pole přehledně zobrazuje obrázek 3.

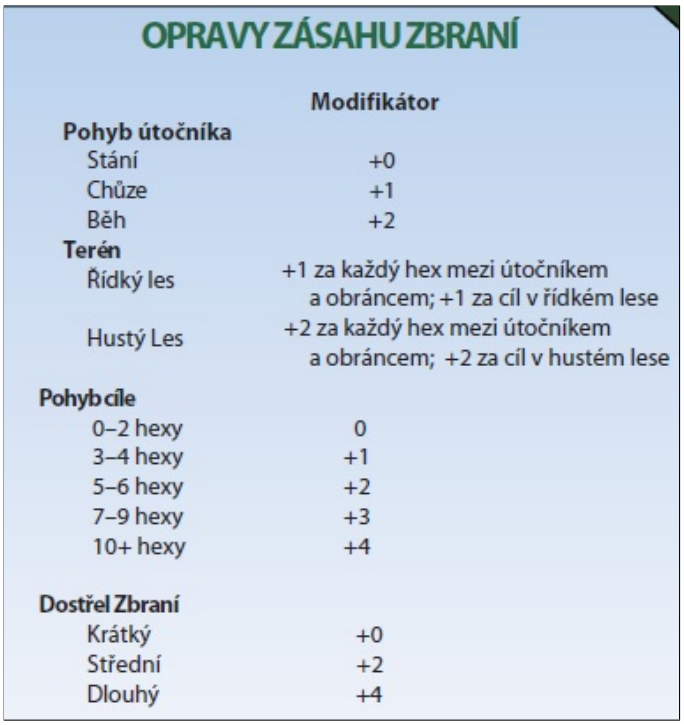


Obrázek 3: Ukázka OBRova palebného pole.

2.6.3 Palba zbraní

Pokud hráčův OBR vidí na svůj cíl a pokud je cíl v útočníkově palebném poli, může na něj vystřelit ze svých zbraní. Kterými zbraněmi bude útočník střílet závisí výhradně na hráči. Pro každou zbraň se pak určí VZZ, do kterého se započítají všechny vlivy, které působí na přesnost střelby. Efekty těchto vlivů se přičtou k ZZZ, který se rovná schopnosti střelby OBRova warriora, podle tabulky na obrázku 4. Aby útočící OBR zasáhl danou zbraní cíl,

musí hráč hodit hodem 2k6 stejně nebo více, než je VZZ pro tuto zbraň. Tento postup se pak opakuje pro každou zbraň, ze které OBR v tomto kole střílí.



OPRAVY ZÁSAHU ZBRANÍ

	Modifikátor
Pohyb útočníka	
Stání	+0
Chůze	+1
Běh	+2
Terén	
Řídký les	+1 za každý hex mezi útočníkem a obráncem; +1 za cíl v řídkém lese
Hustý Les	+2 za každý hex mezi útočníkem a obráncem; +2 za cíl v hustém lese
Pohyb cíle	
0–2 hexy	0
3–4 hexy	+1
5–6 hexy	+2
7–9 hexy	+3
10+ hexy	+4
Dostřel Zbraní	
Krátký	+0
Střední	+2
Dlouhý	+4

Obrázek 4: Tabulka postihů k palbě.

2.6.4 Poškození

Pokud je palba zbraní úspěšná, způsobí zasaženému OBRovi předem stanovené poškození, které odpovídá síle zbraně. Výjimku tvoří zásah raketovým útokem, u kterého se vždy určuje, kolika raketami byl cíl zasažen.
Při každém úspěšném zásahu se následně určí, která část OBRa byla zasažena. Toto se provede hodem 2k6, jehož výsledek odpovídá zasažené části OBRova těla podle tabulky místa zásahu. Poškození, způsobené zásahem, se poté zaznamená do údajů o OBRovi a sice tak, že se od hodnoty pancíře v zasažené oblasti odečte počet bodů poškození, které se rovnají síle zbraně.

2.6.4.1 Raketový útok Střelba z RDD a RKD se řídí dodatečnými pravidly. Při každém úspěšném zásahu se musí navíc ještě určit, kolika raketami byl cíl zasažen, což se provede hodem 2k6. Určení místa zásahu pak probíhá obdobně, jako u ostatních zbraní, pouze s tím rozdílem, že u RKD se určuje místo zásahu pro každou raketu, která zasáhla cíl, zvlášť a u RDD se rakety rozdělí do svazků po pěti (a případně na jeden menší svazek při nedělitelnosti pěti), přičemž se určuje místo zásahu pro každý svazek samostatně.

2.6.5 Zničení OBRa

OBR je vyřazen ze hry v případě, že došlo ke zničení jeho trupu nebo hlavy. Část těla je zničená, pokud hodnota pancíře vnitřní struktury této části klesne na nulu.

3 Programátorská dokumentace

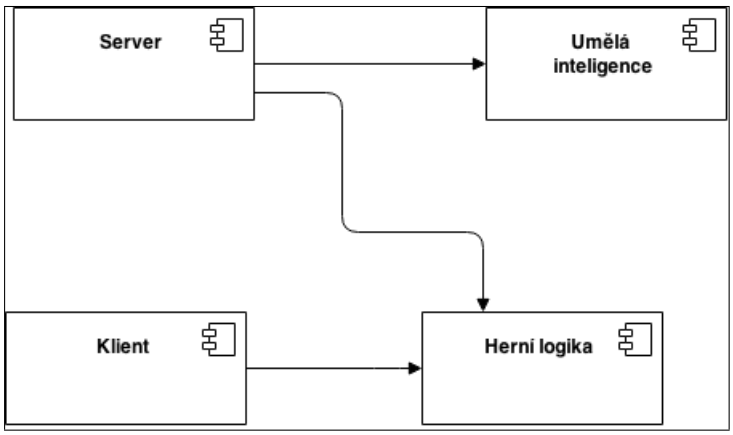
Program je napsán v jazyce Java s podporou JDK 1.8 a využívá také CSV a XML soubory. Lze ho rozdělit na čtyři hlavní části - model, prezentační část, síťová komunikace a editory. V této části se budu zabývat prvními třemi, protože editory budu popisovat v samostatné části později.
Prezentační část je založena na využívání knihovny Swing. Zbylé části programu využívají běžné knihovny z jazyka Java, které definují práci s kolekcemi, streamy, matematickými funkcemi atd.
Program byl vytvořen v IDE Eclipse verze 4.4.1 (Luna). Obrázky, využité v programu, byly vytvořeny v programu Inkscape.

Stručné schéma celého projektu je k vidění na obrázku 5. Na tomto schématu lze vidět, že klient a server jsou oddělení, ale oba mají přístup k herní logice. To proto, že při hře na jednom počítači musí být herní logika k dispozici lokálně, ovšem při hře přes síť nebo proti UI běží herní logika na serveru. Na obrázku 6 je pak vidět způsob komunikace mezi klienty a serverem pomocí protokolu TCP/IP.

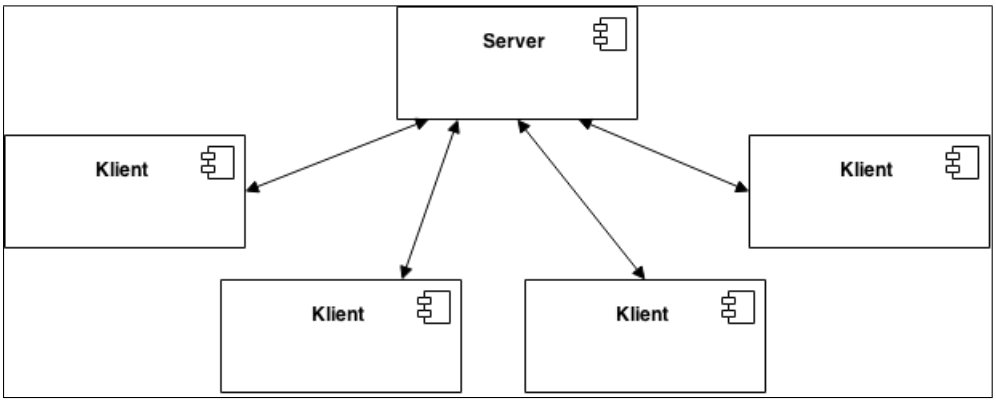
Model hry obsahuje především herní logiku, hráče, OBRy a zbraně. Prezentační část zobrazuje veškeré potřebné informace o hře hráči a mimo jiné mu umožňuje hru ovládat, vytvářet nové hry, připojovat se k existujícím hrám a používat herní editory. Síťová komunikace funguje na architektuře klient-server, ve které mohou klienti vytvářet na serveru hry nebo se k nim prostřednictvím serveru mohou připojit. Server pak zprostředkovává komunikaci mezi hráči a zajišťuje herní logiku.

Během vytváření návrhu a i později během implementace jsem často narážel na problémy spojené s tím, že hra využívá herní mapu složenou z šestiúhelníků. Některé výpočty a mechanismy jsou proto komplikovanější, než tomu je u her se čtvercovou strukturou políček nebo u těch, kde se figurky mohou pohybovat například pouze vpřed (Člověče, nezlob se a obdobné hry). Problematika tvorby her se sítí šestiúhelníků reprezentujících herní plán je velmi podrobně rozpracována na stránkách [2], na jejichž části se budu dále odkazovat v pozdějších částech této práce.

Dalším problémem, na který jsem narazil, bylo zpracování hry pro více hráčů. Desková verze hry je určená pro 2 a více hráčů, ovšem samotná hra je zamýšlená jako souboj dvou soupeřících stran. Pravidla s větším počtem stran vůbec nepočítají. Hráči se tedy vlastně rozdělí na dva týmy a v těchto týmech pak proti sobě bojují. Jedinou možností, jak zpřístupnit hru více než dvěma hráčům najednou by tedy bylo vytvoření týmů z hráčů s tím, že každý z nich bude ovládat jen některé OBRy svého týmu. I tato varianta je však velmi problematická, protože když je na tahu některá ze stran, může provést tah kterýmkoli svým OBRem, se kterým ještě netáhla v dané fázi kola. To znamená, že při větším počtu hráčů v jednom týmu se tito musí domluvit, se kterým OBRem budou provádět tah. Implementace takovéto domluvy by bohužel vedla ke značnému zpomalení průběhu hry, což by mohlo způsobit až nehratelnost při velkých počtech bojujících



Obrázek 5: Schéma projektu.



Obrázek 6: Komunikace mezi klienty a serverem.

OBRů. Hra by se totiž mohla protáhnout na mnoho hodin, což určitě není žádoucí. Z těchto důvodů jsem se nakonec rozhodl pro implementaci hry pouze pro 2 hráče, z nichž každý ovládá všechny OBRy jedné ze soupeřících stran.

Při psaní kódu jsem se snažil pomocí Javadoc komentářů komentovat pokud možno co nejvíce funkcí, bohužel jsem však byl nucen okomentovat pouze některé funkce, které se mi zdály nejdůležitější, protože projekt je poměrně rozsáhlý a komentování všech funkcí by bylo velmi časově náročné. Přesto však doufám, že se mi podařilo dostatečně okomentovat všechny, nebo alespoň převážnou většinu, nejpodstatnější funkce.

Také jsem se snažil dodržovat běžné konvence pro psaní kódu v jazyce Java, i když některé věci jsem psal trochu jiným způsobem, na který jsem zvyklý a díky kterému je pro mě kód přehlednější a čitelnější. Toto se týká asi především používání Allmanovy notace místo běžně doporučované 1TBS notace.

3.1 Model hry

Model hry obsahuje základní herní logiku. Při jeho tvorbě jsem se především snažil o to, aby byl použitelný jak pro lokální verzi hry hratelnou na jednom počítači, tak i pro online verzi. Také jsem chtěl herní logiku co nejvíce oddělit od prezentační části programu.

3.1.1 Balík game

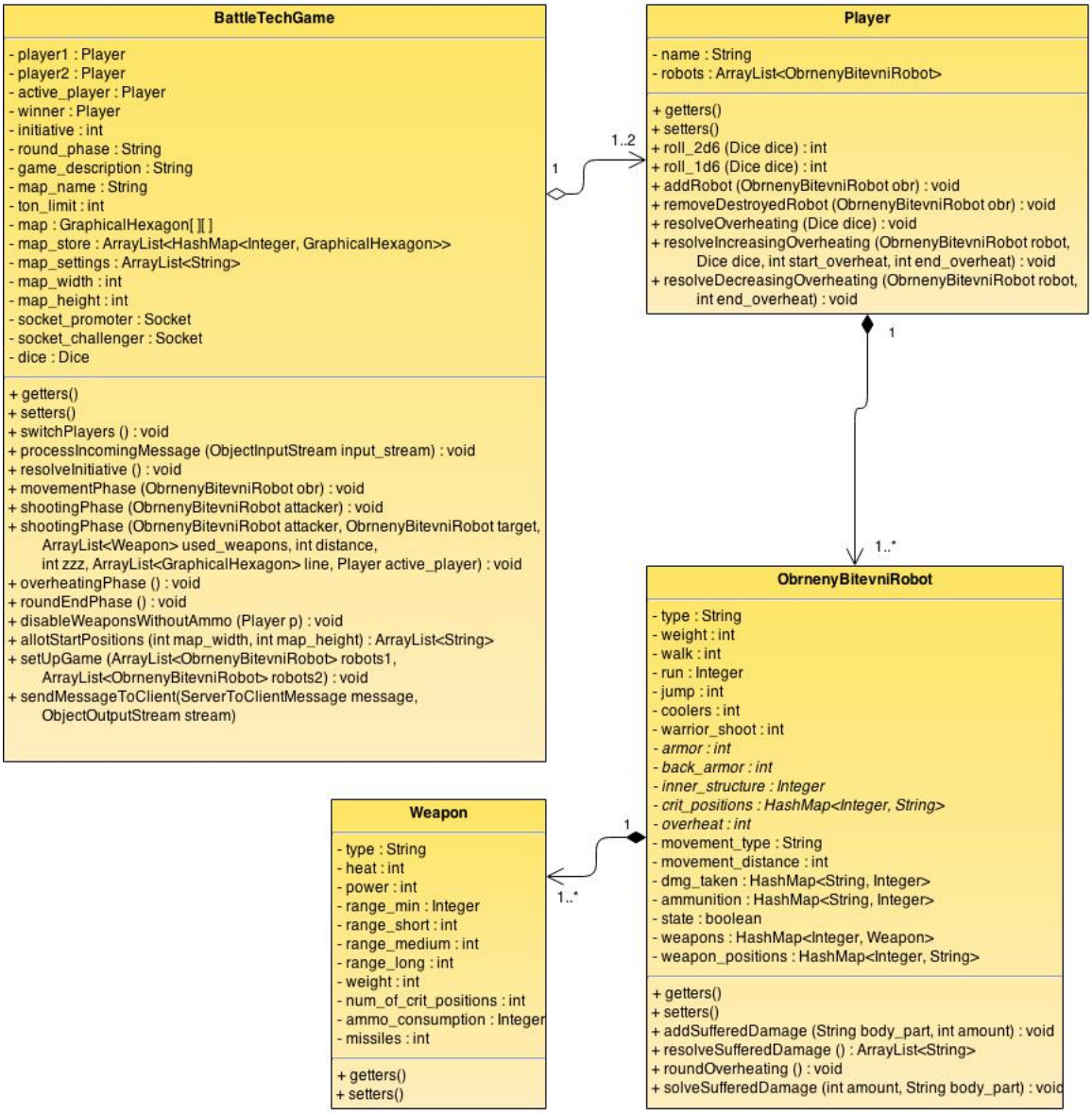
V balíku *game* se nacházejí třídy, které tvoří model hry. Nejdůležitějšími třídami v balíku jsou třídy BattleTechGame, Player, ObrnenyBitevniRobot a Weapon, které jsou znázorněny na obrázku 7. Kromě nich obsahuje balík několik dalších tříd, sloužících jako pomocné třídy, které řeší různé výpočty nebo poskytují méně významné funkce.

BattleTechGame Tato třída řídí průběh hry, udržuje si informace o fázi herního kola a o hráči na tahu, zpracovává tahy hráčů a v případě potřeby rozesílá aktuální informace o stavu hry klientům. Také má informace o aktuálním stavu všech robotů.
Hlavními metodami této třídy jsou ty, které řídí průběh celé hry. Funkce *resolveInitiative()* určuje na začátku každého kola hráče, který bude mít iniciativu v tomto kole. Pohyb robota zpracovává *movementPhase()* a robotovu střelbu zase dvě metody *shootingPhase()*, jedna pro střelbu na cíl a druhá pro variantu, kdy OBR nestřílí. Po ukončení fáze střelby dojde k výpočtu přehřívání OBRů metodou *overheatingPhase()* a na závěr funkce *roundEndPhase()* dokončí herní kolo.

Player Třída, která představuje hráče. Obsahuje informace o hráči a především seznam jeho OBRů.
Metody *roll_2d6()* a *roll_1d6()* simulují hody kostkami/kostkou. Dále má k dispozici metody pro výpočet přehřívání OBRů a zaznamenání příslušných důsledků, které z případného přehřátí vzniknou.

ObrnenyBitevniRobot Tato třída uchovává všechny informace o OBRovi, o jeho aktuálním stavu a o schopnostech jeho warriora.
Pomocí metod *resolveSufferedDamage()* a *solveSufferedDamage()* dojde k zaznamenání poškození, které OBR utrpěl v daném kole a které si předtím připravil ke zpracování metodou *addSufferedDamage()*
Jsou zde také velké možnosti budoucího rozšíření. Úplná verze hry totiž obsahuje možnost kritických zásahů, které mohou mít velký vliv na stav OBRa, nebo třeba více atributů u warriora. Možnostem budoucího rozšíření či vylepšení programu se však budu více věnovat v závěru práce.

Weapon Reprezentuje zbraň, kterou může mít OBR ve své výbavě, a obsahuje všechny důležité informace o této zbrani (například její sílu nebo počet jednotek tepla, které vyprodukuje při střelbě).



Obrázek 7: Třídní diagram hlavních tříd herní logiky

Poznámka 3.1 Na obrázku jsem do diagramu pro větší přehlednost zahrnul pouze hlavní třídy a u třídy ObrnenyBitevniRobot jsem některé atributy sdružil do jednoho. Takovéto atributy jsou vyznačeny kurzívou.

Zajímavé metody Metoda *Calculations.hexLinedraw()* spočítá souřadnice všech hexů, které leží na spojnici mezi libovolnými dvěma hexy. Nejdříve převede dvourozměrné souřadnice na třírozměrné dopočítáním třetí souřadnice z prvních dvou a poté vypočítá třírozměrné souřadnice všech hexů na hledané spojnici. Pro výpočet nejdříve potřebujeme znát vzdálenost krajních hexů n, kterou spočítáme metodou *Calculations.cubeDistance()*. Celkový počet hexů na spojnici bude $n + 1$. Funkce *Calculations.cubeLinedraw()* pak pro každý hex na spojnici spočítá metodou lineární interpolace (viz [3]) jeho souřadnice. Konkrétní použitý vzorec vypadá takto (více viz [4]):

$$\begin{aligned} Cx &= Ax + (Bx - Ax) \times k \\ Cy &= Ay + (By - Ay) \times k \\ Cz &= Az + (Bz - Az) \times k \end{aligned} \tag{1}$$

kde A, B představují krajní hexy spojnice, C je hex ležící na spojnici, x, y, z jsou souřadnice hexu a k je koeficient vypočítaný vzorcem

$$\frac{1}{n \times index\_hexu}$$

kde $index\_hexu$ je číslo od 0 do n.
Takto postupně získá souřadnice všech hexů na spojnici. Tyto souřadnice pak na závěr ještě převede zpět do dvourozměrných souřadnic, se kterými se pak může dále pracovat (například při ověření, zda OBR vidí na svůj cíl).

Balík *game* také obsahuje třídu Dice, která simuluje hrací kostku a má metody pro hod jednou i dvěma kostkami, které jsou ve hře často potřeba. Pro věrnější reprezentaci skutečného hodu dvěma kostkami jsem se rozhodl implementovat tuto metodu tak, že se použije součet dvou hodů jednou kostkou, přestože by se to samozřejmě dalo vyřešit vygenerováním náhodného čísla od 2 do 12. Takový postup by však změnil pravděpodobnosti jednotlivých čísel, že padnou, protože generování náhodného čísla pomocí pseudonáhodných generátorů jazyka Java pokrývá rovnoměrně celý zadaný rozsah. Z toho důvodu by pravděpodobnost, že padne číslo 2, byla přibližně stejná jako pravděpodobnost, že padne číslo 7, což rozhodně neodpovídá skutečnosti a narušilo by to herní mechanismus.

Při implementaci metody *ObrnenyBitevniRobot.solveSufferedDamage()* jsem narazil na nepříjemnou situaci. Každý OBR má totiž osm částí těla, z nichž každá má dvě nebo i tři hodnoty pancíře, které jsou uchovávány v různých atributech. Navíc při zničení vnějšího pancíře se poškození přesouvá do vnitřní struktury a po zničení vnitřní struktury dané části těla se poškození buď přenáší do jiné části těla, nebo dojde ke zničení OBRa. Tato situace mě nejdříve vedla k velkému množství podmínek, v nichž se pak z velké části vykonával téměř stejný kód, který se pouze lišil v názvech metod. Rozhodl jsem se, že to musí jít řešit nějak obecněji, protože použité metody byly téměř výhradně metody get a set a lišily se pouze v tom, jakou část těla zpřístupňovaly. Naštěstí v Javě existuje balík *java.lang.reflect*, který poskytuje třídy přesně řešící můj problém. Pomocí tohoto balíku

jsem mohl obecnou část vyřešit mnohem jednodušeji bez zbytečného opakování téměř totožného kódu (viz výpis 1).

```
public void solveSufferedDamage (int amount, String body_part) throws
      IllegalAccessException, IllegalArgumentException, InvocationTargetException
{
   String getter = "get" + body_part;
   String setter = "set" + body_part;
   Method methodGetter;
   Method methodSetter;
   try {
      methodGetter = this.getClass().getMethod(getter);
      methodSetter = this.getClass().getMethod(setter, int.class);
   } catch (Exception e) {
      methodGetter = null;
      methodSetter = null;
      e.printStackTrace();
   }
   methodSetter.invoke(this, ((int)methodGetter.invoke(this)) - amount);
   int armor_value = (int) methodGetter.invoke(this);
   .
   .
}
```

Výpis 1: Ukázka Java Reflection API

3.2 Prezenační část

Prezentační část hry zobrazuje herní součásti hráči a umožňuje mu hru ovládat. Přestože je vzhled aplikace určitě velmi důležitý, použil jsem spíše jednoduchou grafiku. Doufám však, že tato grafika je dostačující pro dobrou hratelnost. V každém případě je to ale jedna z věcí, kterou by určitě bylo dobré do budoucna vylepšit.

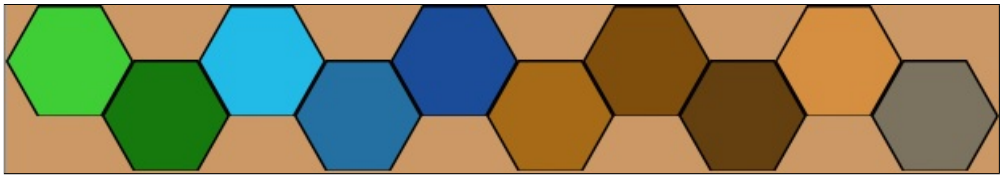
3.2.1 Balík GUI

V balíku *GUI* se nachází třídy pro grafické znázornění hry. Hlavní třídou je BattleTechGUI, která obsahuje kompletní grafické uživatelské rozhraní a využívá při tom třídy RobotImage a GraphicalHexagon, která ještě využívá třídu Hexagon.

BattleTechGUI Hlavní třída grafického rozhraní. Zobrazuje veškeré informace o hře a umožňuje přístup ke všem funkcím programu. Obsahuje grafické prvky, které nabízí knihovna Swing a metody pro navázání spojení se serverem a následnou komunikaci.

RobotImage Představuje figurku OBRa na hracím plánu. Základní barvy jsou černá a bílá pro odlišení OBRů jednotlivých stran. Pokud je OBR vybrán hráčem pro tah, figurka se zbarví do žluta a OBR označený jako cíl útoku bude mít figurku červené barvy. Po ukončení tahu se barva vždy vrátí do původního stavu podle příslušnosti k jedné či druhé straně.

GraphicalHexagon a Hexagon Třída Hexagon obsahuje informace o políčku herního plánu - především jeho souřadnice a typ terénu. Také umožňuje určit souřadnice sousedních políček. Třída GraphicalHexagon pak doplňuje ke třídě Hexagon grafické znázornění políčka na mapě. Také definuje tvar políčka pomocí metody *contains()* a jeho barvu pomocí metody *paintComponent()*, která na pozadí komponenty vykreslí příslušný obrázek ze složky */Obrazky*. Všechny možné typy terénu jsou zobrazeny na obrázku 8.



Obrázek 8: Ukázka všech možných typů terénu.

Poznámka 3.2 Zleva se jedná o řídký les, hustý les, vodu hloubky 0, 1 a 2, hory úrovně 1, 2 a 3, lehký terén a těžký terén.

3.2.2 Grafické uživatelské rozhraní

Hlavní obrazovka je rozdělena na pět částí. Uprostřed je zobrazena herní mapa, napravo panel pro zobrazení informací o OBRovi, nalevo panel pro ovládací prvky, nahoře jsou informace o stavu hry a dole je textové pole, do kterého se vypisuje průběh hry.

Herní mapa Herní mapa se skládá z šestiúhelníků, což poněkud komplikuje vykreslování. Je potřeba umístit komponenty tak, aby se šestiúhelníky dotýkaly hranami svých sousedů. Pro vykreslení mapy používám cyklus uvnitř cyklu, přičemž pro každý šestiúhelník spočítám jeho souřadnice uvnitř mřížky a také jeho pozici uvnitř zobrazovacího panelu. Souřadnice pozice komponenty se počítají pomocí tří vzorců. Hodnota x je odvozena od šířky komponenty, která odpovídá dvojnásobku délky hrany. Vzorec vypadá následovně:

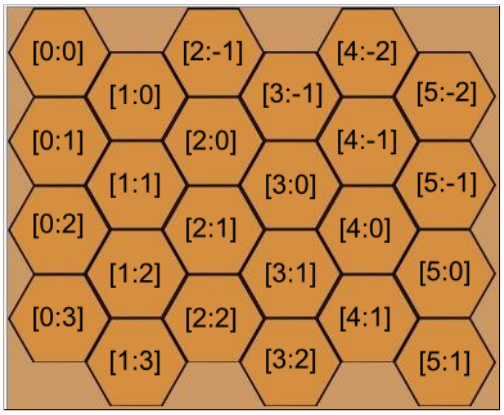
$$x = \left(\left(\frac{h \times 2}{4}\right) \times 3\right) \times i \tag{2}$$

kde h je délka hrany šestiúhelníku a i je iterace cyklu.
Pro y je nutné mít dva vzorce, protože každý druhý sloupec šestiúhelníků je jakoby posunutý o kousek dolů. Vypadají takto:

$$\begin{aligned} y &= v \times j \\ y &= v \times j + \tfrac{v}{2} \end{aligned} \tag{3}$$

kde v je výška šestiúhelníku a j je iterace cyklu.
Další problém, který vyplývá z mapy tvořené šestiúhelníky, je systém souřadnic uvnitř mřížky těchto políček. Při vytváření návrhu jsem narazil na tři možné systémy, které lze

použít. Detailně jsou všechny popsány na stránkách [5]. Já jsem se rozhodl použít tzv. axiální systém, který umožňuje jednodušší algoritmy pro výpočty uvnitř mřížky a přitom není nutné mít tři souřadnice. Číslování jednotlivých políček pak vzniká poměrně jednoduše. V levém horním rohu je políčko $[0:0]$. Souřadnice x určuje sloupec a její hodnota roste s každým dalším sloupcem o 1. Souřadnice y se liší v tom, že každé druhé horní políčko má hodnotu o 1 menší než horní políčko předchozího sloupce. Čísla řádků pak také rostou vždy o 1 oproti políčku o řádek výš. Ukázka číslování políček je vidět na obárzku 9.



Obrázek 9: Ukázka systému souřadnic herního plánu.

Podrobněji se budu uživatelskému rozhraní věnovat v uživatelské dokumentaci v pozdější části této práce, kde bude popsáno i základní ovládání programu.

3.3 Síťová komunikace

Síťová komunikace je implementována jako klient-server architektura a umožňuje hráčům hrát hru přes internet. Jelikož herní portál, na který měla být tato hra původně nasazena, nebyl v době psaní této práce ještě v provozu, je v tuto chvíli hra nastavená pro testovací účely pro hru přes síť pouze lokálně na jednom počítači. Veškerá funkcionalita je však v práci zahrnuta, především síťová komunikace se serverem a vytváření nových instancí hry. Domnívám se, že pro nasazení na herní portál by nebylo potřeba dělat mnoho změn, protože program obsahuje i implementaci vlastního jednoduchého serveru.
Kódy potřebné pro síťovou komunikaci jsou v balících *server* a *networkCommunication*.

3.3.1 Balík server

V tomto balíku jsou dvě důležité třídy - *Server* a *ServerMaster*. Tyto třídy zajišťují hlavní funkcionalitu serveru, umožňují připojování klientů a spojují hráče pro hru přes síť.

Server Tato třída reprezentuje server. Běží v nekonečné smyčce a její úkol je zpracovávat připojení klientů. Pokud se připojí k serveru nový klient, vytvoří *Server* nové vlákno třídy *ServerMaster*, které pak dále komunikuje s klientem, zatímco *Server* bude znovu očekávat připojení dalšího klienta.
Obsahuje popisy her a také vytvořené hry.

ServerMaster Funguje jako jakýsi správce. Komunikuje s klientem a umožní mu buď vytvořit novou hru nebo se k již vytvořené hře připojit. Při vytváření hry založí na serveru novou instanci hry, přidá popis nové hry do seznamu popisů a rovnou připojí klienta k této hře. Při připojení ke hře nejdříve klientovi pošle seznam popisů existujících her, ze kterých si klient vybere a na základě jeho výběru ho připojí k vybrané hře. Instance hry se při připojení druhého hráče spustí v novém vlákně a samotná hra může začít.

3.3.2 Balík networkCommunication

Tento balík obsahuje rozhraní *ClientToServerMessage* a *ServerToClientMessage* a řadu tříd, které slouží ke komunikaci mezi serverem a klientem. Každá z těchto tříd představuje typ zprávy, které si mohou server a klient mezi sebou posílat. Takováto zpráva je poslána přes *ObjectOutputStream* a na druhé straně načtena z *ObjectInputStream*. Po načtění je zpráva zpracována v závislosti na tom, o jakou třídu se jedná.

3.4 Umělá inteligence

Tvorba UI pro mě byla velice zajímavá, protože s vývojem něčeho takového jsem se ještě nesetkal. Vzhledem k tomu, že se má jednat pouze o jednoduchou UI, rozhodl jsem se založit většinu akcí na základě pseudonáhodně generovaných čísel. Třída *AIOponent*, která představuje UI, se nachází v balíku *artificialIntelligence*. Vzhledem k tomu, že hlavní zamýšlený účel UI je schopnost nahradit hráče, který se odpojí od online hry, implementoval jsem ji jako klienta, který se připojí k serveru a následně s ním komunikuje stejným způsobem, jako by to byl normální hráč. Z pohledu serveru se tedy vůbec nic nemění, dostává ve stejnou chvíli stejné typy zpráv, jako by na druhé straně byl běžný uživatel. Přesto bych však do budoucna rád udělal i verzi UI pro hru pouze na lokálním počítači bez nutnosti připojení k serveru. Tato verze by pak mohla sloužit k seznámení se s hrou a naučení základů hry bez nutnosti připojení k serveru.

Aby mohla UI simulovat skutečného hráče, musí umět provádět všechny tahy a rozhodnutí, které provádí hráč. To znamená, že musí umět vybrat pro svého OBRa cíl, v případě potřeby se přesunout tak, aby na něj bylo možno střílet a samozřejmě zvolit, kterými zbraněmi bude OBR pálit.

Volba cíle je jednoduchá. UI každému svému OBRu určí jeho cíl tak, že spočítá vzdálenost od OBRa postupně ke všem soupeřovým OBRům a ten, který je nejblíž, se stává cílem. Cíl OBRa se pak už nemění v daném kole a určuje se znovu až na začátku příštího

kola.

Nejnáročnější bylo zajistit, aby byla UI schopná pohybovat svými OBRy po mapě nějakým rozumným způsobem. Pro určení pohybu slouží metody *moveRobot()* a *findNewPosition()*. Pro výpočet nové OBRovy pozice jsem nejdříve musel určit nějaký optimální cílový stav, do kterého se OBR bude snažit dostat. Jako optimální jsem nakonec určil vzdálenost šest hexů od cíle. Tato vzdálenost mi přišla optimální, protože většina zbraní s krátkým dostřelem má vzdálenost šest hexů jako horní hranici středního dotřelu a řada zbraní s dlouhým dostřelem má tuto vzdálenost jako krátký dostřel. Navíc některé zbraně s minimálním dostřelem mají právě tuto vzdálenost jako minimální dostřel. Tím pádem je vzdálenost šesti hexů optimálním kompromisem, který vyhovuje většině zbraní a umožňuje střelbu s malým nebo žádným postihem k palbě. Samozřejmě nestačí pouze to, aby se OBR dostal na vzdálenost šest hexů od svého cíle, ale musí na něj zároveň i vidět. Pokud se OBR nemůže se svými BP dostat do optimálního cílového stavu, přijde alespoň tak blízko k cíli, jak jen může, s tím, že na cíl musí vidět.

Fáze střelby pak opět není nijak složitá. Pokud OBR může střílet na svůj cíl, u každé zbraně, ze které může střílet, se UI rozhodne, jestli z ní vystřelí nebo ne. Toto rozhodování funguje na základě pseudonáhodného generování čísel v metodě *flipCoin()*, která simuluje hod mincí.

moveRobot() Tato metoda určí pohyb OBRa. Je rozdělena na dvě části. Pokud je OBR ve vzdálenosti šest a méně hexů od svého cíle, nikam se nepřesouvá a pouze pokud nemá cíl ve svém zorném poli, natočí se na svém hexu tak, aby svůj cíl viděl. Pokud je od cíle dále než šest hexů, pomocí metody *findNewPosition()* vypočítá novou OBRovu pozici. Pro zjednodušení UI pohybuje svými OBRy vždy chůzí.

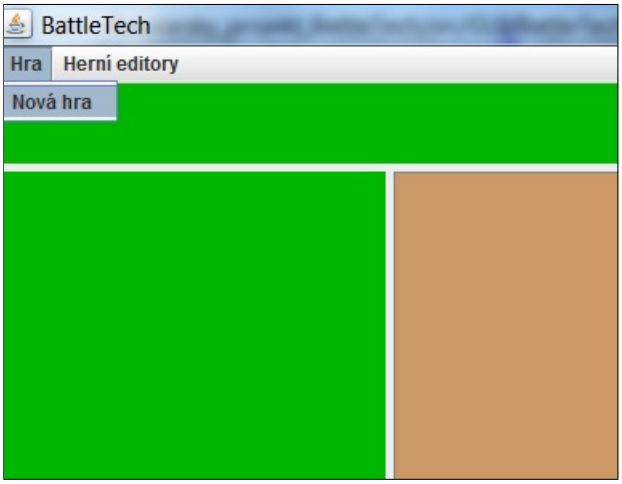
findNewPosition() Tato metoda vytvoří pro nalezení nové OBRovy pozice graf, který pak následně prochází algoritmem prohledávání do šířky (viz [6]). Uzel grafu je reprezentován souřadnicemi hexu uvnitř mřížky a natočením OBRa na tomto hexu. Tyto údaje jsou ukládány do instancí statické třídy *HexNode*. Na začátku je vytvořen kořen grafu, který odpovídá počáteční pozici OBRa. Tento uzel je přidán do fronty, ze které algoritmus postupně bere jednotlivé uzly a následně je zpracovává, dokud nevyprázdní frontu nebo se nedostane do optimálního cílového stavu. Pokud aktuální uzel grafu není cílovým, vytvoří se z něj vždy čtyři nové uzly, které odpovídají OBRovým možnostem pohybu z aktuálního stavu. OBR se totiž může vždy pohnout pouze čtyřmi způsoby (vpřed, vzad a otočení doprava či doleva). Každý uzel, který byl zpracován, se označí jako navštívený. Pokud se v grafu znovu objeví stejný uzel, nebude se nijak zpracovávat a ihned se přejde na další uzel z fronty.
Pokud algoritmus zpracovává uzel, ve kterém již OBR nemá žádné BP, uloží se tento uzel jako možné řešení pro případ, že algoritmus nenajde optimální cílový stav. V takovém případě použije jako novou pozici OBRa uzel, ve kterém byl OBR nejblíže k cíli.

4 Uživatelská dokumentace

V této části bude popsána aplikace z pohledu uživatele. Bude zde popis ovládání a také orientace v aplikaci, obojí v případě potřeby doplněno názornými ukázkami přímo z aplikace. Uživatelské rozhraní i veškeré funkce a ovládání hry bylo vytvářeno se snahou o maximální jednoduchost pro zajištění pokud možno snadného ovládání a intuitivního přístupu ke všem funkcím. U méně intuitivních prvků ovládání byla přidána nápověda uvnitř hry, která by měla uživateli stručně sdělit, jak dosáhnout požadovaného cíle.

4.1 Uživatelské menu

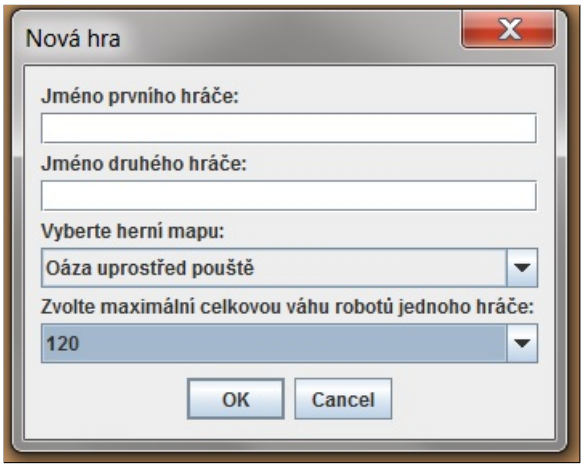
Uživatelské menu je klasické a jednoduché se snadnou možností budoucího rozšíření o nové funkce. Náhled menu je vidět na obrázku 10, kde je také vidět, jak spustit novou hru.



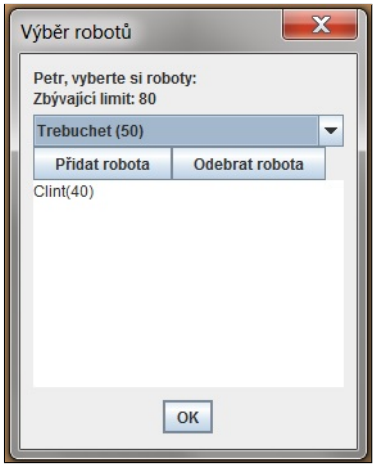
Obrázek 10: Ukázka základního menu.

Navigace vytvářením nové hry je znázorněna na obrázcích 11 a 12. Při volbě typu hry má hráč možnost volby mezi třemi možnostmi. *Hra na jednom počítači* je jediná, pro jejíž hraní není zapotřebí připojení k serveru. V tomto typu hry hrají hráči na jednom počítači. Při *hře přes internet* je pochopitelně nutné připojení k serveru, který hráče spojí s jeho protivníkem. U *hry proti počítači* je nutnost připojení spíše dočasnou podmínkou. Do budoucna by určitě bylo vhodné rozšířit tuto možnost i o lokální variantu, ve které by UI klient běžel lokálně na počítači hráče a hra by tím pádem nevyžadovala žádné připojení. Ve všech typech hry je nutné zvolit si své OBRy podle maximálního možného součtu jejich hmotností. Ukázka okna s touto volbou je zobrazena na obrázku 13.

Obrázek 11: Výběr typu nové hry.



Obrázek 12: Nastavení hry.

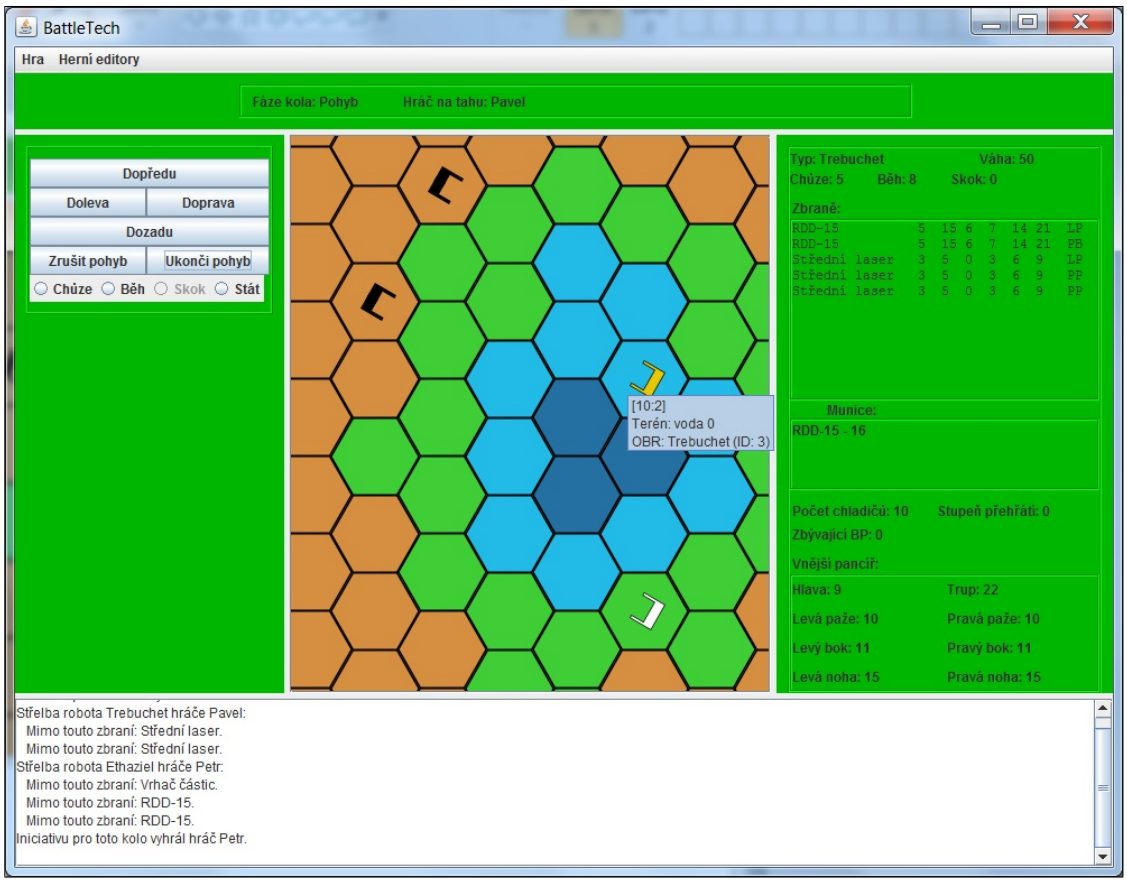


Obrázek 13: Volba OBRů.

4.2 Ovládání hry

Po spuštění hry se uprostřed obrazovky objeví herní mapa včetně figurek OBRů. V horním panelu se objeví informace o hře, konkrétně fáze kola a který hráč je na tahu. U online verze hry se zde navíc zobrazí jméno hráče, který hraje na tomto zařízení.
Levý a pravý panel se mění podle toho, který OBR je označený, či podle toho, jaká je fáze herního kola. Herní obrazovka je vidět na obrázku 14. Jak je z obrázku vidět, herní mapa poskytuje informace o jednotlivých políčcích. Při najetí myší nad políčko se zobrazí jeho souřadnice, typ jeho terénu a pokud je na políčku OBR, pak i jeho název a ID. Ve spodní části obrazovky se nachází záznam herních událostí.

Vždy, když je hráč na tahu, musí si vybrat jednoho svého OBRa, se kterým v této fázi kola ještě netáhl, a provést s ním příslušné akce. Výběr OBRa se provede jednoduše kliknutím na jeho figurku na mapě. Po zvolení OBRa se v pravé části obrazovky zobrazí informace o tomto OBRovi a v levé části budou k dispozici tlačítka potřebná k tahu v této fázi kola.
Ve fázi pohyb bude ovládací panel vypadat jako na obrázku 15. Hráč musí vždy nejdříve určit, jakým způsobem se OBR bude pohybovat. Pokud nechce, aby se OBR pohyboval, musí zvolit možnost *Stát*. Jakmile je hráč spokojen s pozicí svého OBRa, ukončí jeho po-



Obrázek 14: Spuštěná hra.

hyb tlačítkem *Ukonči pohyb*. Do té doby může hráč s OBRem pohybovat po mapě zcela libovolně. Může měnit typ pohybu, může dosavadní pohyb zrušit a začít znovu nebo může i zvolit jiného svého OBRa a provést tah s ním. Pohyb *chůzí* a *během* se uskutečňuje pomocí čtyř tlačítek pro pohyb na ovládacím panelu. Pohyb *skokem* se liší v tom, že hráč nemusí postupně přesouvat svého OBRa na cílový hex, ale jednoduše pravým tlačítkem klikne na tento hex a OBR se tam přesune. Poté může hráč OBRa ještě libovolně natočit pomocí tlačítek *Doleva* a *Doprava*. Pokud s pozicí svého OBRa není spokojen, může hráč pohyb zrušit a začít s ním znovu.

Po ukončení fáze pohyb přejde hra do fáze *Střelba zbraněmi*. Jakmile k tomu dojde, změní se i ovládací panel do podoby ilustrované obrázkem 16. Hráč musí opět zvolit svého OBRa, se kterým bude provádět tah, a protože se jedná o střelbu, musí zvolit i cíl. Pokud nechce střílet, musí označit volbu *Nestřílet*. Jakmile hráč zvolí cíl, což se provádí kliknutím pravým tlačítkem myši na soupeřova OBRa, může si kliknutím na *Vybrat zbraně* zobrazit zbraně svého OBRa, kterými může vystřelit na určený cíl. Zbraň, která bude použita, je označena zaškrtnutým políčkem vedle svého názvu. Jakmile hráč ukončil volbu zbraní a je spokojen s výběrem cíle i s výběrem zbraní, může ukončit tah tlačítkem *Ukon-*

čit střelbu. Do té doby opět může libovolně měnit výběr svého OBRa, výběr cíle i výběr zbraní.

Obrázek 15: Ovládací panel ve fázi pohyb.

Obrázek 16: Ovládací panel ve fázi střelba zbraněmi.

4.3 Hra přes síť

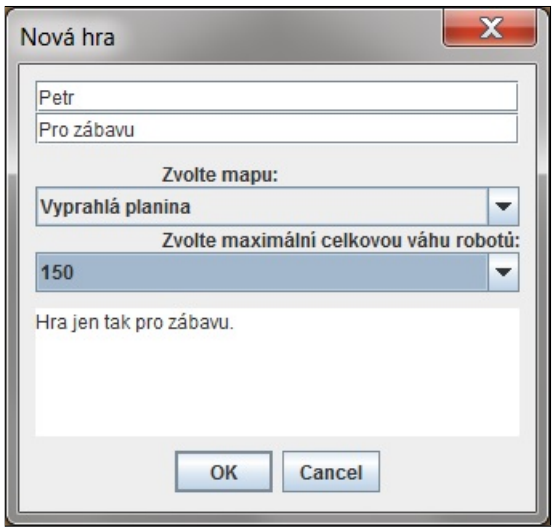
Při hře přes síť je potřeba, aby zakládající hráč vyplnil kromě svého jména a mapy také název a popis hry, aby se podle těchto údajů mohl ke hře někdo připojit. Popis hry může například sloužit jako vzkaz pro protivníka, pokud jsou hráči spolu předem domluveni, aby věděl, ke které hře se připojit. Jednoduchá ukázka vytváření nové hry přes síť je zachycena na obrázku 17. Po vyplnění všech potřebných údajů je hra zaznamenána na serveru a zakládající hráč už musí jen počkat, až se někdo připojí.

Chce-li se hráč připojit k nějaké hře, dostane v případě úspěšného připojení k serveru seznam her, ke kterým se může připojit (ukázka na obrázku 18). U každé hry vidí její název a popis. Jakmile vybere hru a odešle volbu na server, bude připojen k vybrané hře, server spustí nové vlákno s hrou a oba klienti dostanou informace pro nastavení hry a hráči můžou začít hrát.

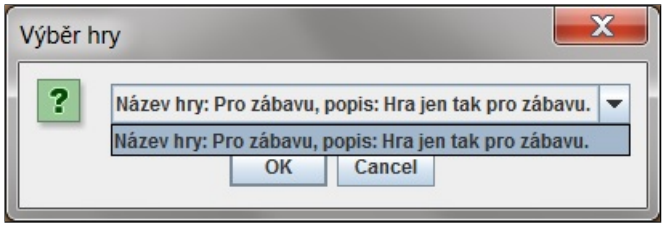
Pokud se nepodaří navázat spojení se serverem, místo další volby se zobrazí odpovídající oznámení o chybě.

4.4 Spuštění hry

Pro spuštění aplikace je potřeba mít nainstalovanou verzi Java Runtime Environment 8, případně nejnovější verzi. Doporučený operační systém pro spuštění je operační systém Windows.



Obrázek 17: Vytvoření hry přes síť.



Obrázek 18: Připojení ke hře přes síť.

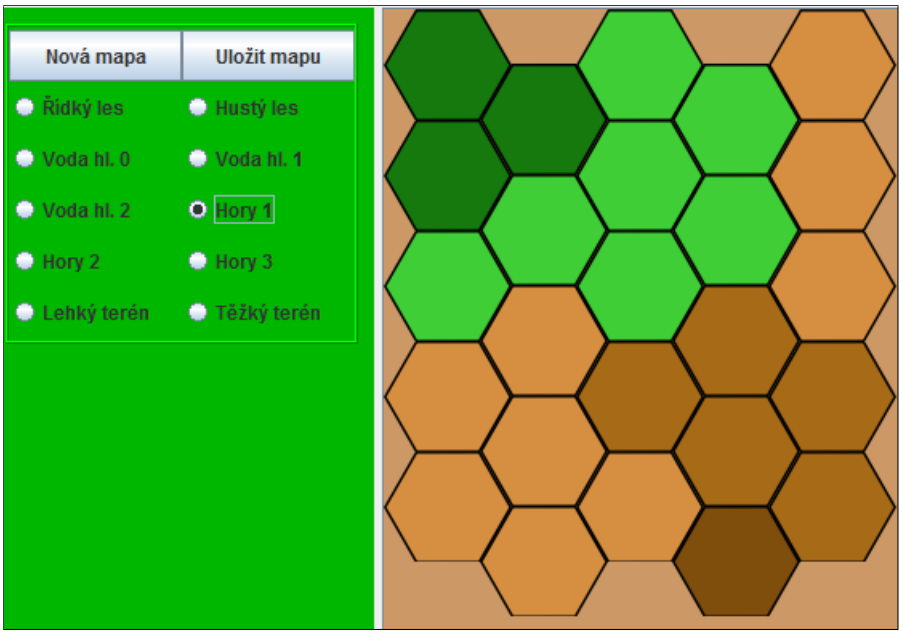
Ke spuštění hry slouží .jar soubor s názvem BattleTech, který spustí klientskou aplikaci. Pro hru přes síť nebo proti UI je třeba spustit ještě server, k čemuž pro testovací účely slouží .jar soubor BattleTech Server.

5 Herní editory

Herní editory, které jsou obsaženy ve hře, byly velmi zajímavým zpestřením při vývoji hry a doufám, že budou také velmi zajímavým bonusem pro uživatele. Tvorba herních plánů není nic tak zvláštního, ale myslím, že není příliš běžné, aby si hráč mohl sestavit vlastní herní postavy a hrát s nimi, aniž by tím narušil herní rovnováhu nebo získal nějakou výhodu. BattleTech však umožňuje sestavení vlastních OBRů podle pevně daných pravidel, které byly uplatněny i při návrhu OBRů obsažených ve hře. Hráč si tedy může vytvořit vlastní OBRy, kteří budou zapadat do světa BattleTechu a nijak nenaruší herní rovnováhu.

5.1 Editor herních plánů

Editor herních plánů je jednoduchý nástroj, který umožní vytvoření vlastních herních map, na kterých potom lze normálně hrát. Po spuštění se objeví na levé straně obrazovky ovládací panel jako na obrázku 19. Po kliknutí na *Nová mapa* a zadání výšky a šířky se uprostřed obrazovky objeví nový herní plán, na kterém mají všechny hexy přednastavený terén na *lehký*. Pomocí přepínání mezi nástroji v ovládacím panelu pak může uživatel libovolně měnit terén kteréhokoli hexu pouhým kliknutím levým tlačítkem myši podobně, jako tomu je na obrázku 19. Když je uživatel se vzhledem nové mapy spokojen, lze mapu uložit tlačítkem *Uložit mapu*. Po zadání jména mapy je mapa uložena a je možno ji zvolit při zakládání nové hry.



Obrázek 19: Editor herních plánů.

5.2 Editor OBRů

Konstrukce OBRů je velmi komplexní proces, sestávající z následujících jedenácti kroků:

1. Určení celkové váhy
2. Určení výkonu reaktoru
3. Zabudování ovládacích prvků
4. Určení váhy vnitřní struktury
5. Určení parametrů skoku
6. Zabudování chladičů
7. Zabudování pancíře
8. Zabudování zbraní a munice
9. Dokončení tabulky kritických zásahů
10. Vyznačení hodnot pancíře
11. Dokončení palubní karty

Některé operace lze provést automaticky bez nutnosti interakce s uživatelem, přesto však musí uživatel určit řadu věcí. Pro usnadnění této činnosti ho editor provede několika formuláři, které mu umožní snadno a jednoduše definovat všechno potřebné.

5.2.1 Určení celkové váhy

Váha OBRa může být 10 až 100 tun a narůstá vždy po pěti tunách. Hráč si musí zvolit celkovou váhu svého OBRa, do které se pak musí vejít veškeré OBRovo vybavení.

5.2.2 Určení výkonu reaktoru

Každý OBR má jeden reaktor, který ho pohání. Váha reaktoru závisí na výkonu reaktoru, který se určí vzorcem *váha robota* $\times$ *požadovaná rychlost chůze* = *výkon reaktoru*. To znamená, že uživatel určí požadovanou rychlost chůze a aplikace pak sama vypočítá výkon a váhu reaktoru. Jeho váha se pak odečte od celkové váhy OBRa.

5.2.3 Zabudování ovládacích prvků

Každý OBR musí být vybaven kokpitem a gyroskopem, aby byl ovladatelný a držel rovnováhu. Kokpit váží vždy 3 tuny. Váha gyroskopu se vypočítá tak, že se výkon reaktoru vydělí stem a výsledek se zaokrouhlí nahoru. Tolik tun pak váží gyroskop OBRa. Tato část probíhá automaticky, aplikace obě hodnoty odečte od zbývající váhy OBRa.

5.2.4 Určení váhy vnitřní struktury

Váha vnitřní struktury odpovídá desetině celkové váhy OBRa. I tato část probíhá automaticky a nevyžaduje žádnou interakci s uživatelem. Váha vnitřní struktury se opět odečte od zbývající váhy OBRa.

5.2.5 Určení parametrů skoku

OBRa lze vybavit zdvižnými tryskami, které mu umožní pohyb skokem. Každá tryska přidá 1 BP skoku, avšak není možné zkonstruovat OBRa, který má více BP skoku než BP chůze. Váha trysky závisí na celkové hmotnosti OBRa (viz tabulka 1) a jejich celková hmotnost je odečtena od zbývající váhy OBRa.

Váha OBRa	**Váha zdvižné trysky**
10-55	0,5 tuny
60-85	1 tuna
90-100	2 tuny

Tabulka 1: Tabulka zdvižných trysek.

5.2.6 Zabudování chladičů

Přestože je každý OBR vybaven automaticky deseti chladiči, většina OBRů jich potřebuje víc. Proto se může hráč rozhodnout, že svému OBRovi přidá přídavné chladiče, jejichž váha je odečtena od zbývající váhy OBRa. Každý přídavný chladič váží jednu tunu.

5.2.7 Zabudování pancíře

Pancíř chrání OBRa před poškozením. Každá tuna pancíře představuje 16 bodů pancíře, které lze rozdělit mezi části OBRova těla. Samotné rozdělení se provádí v pozdější fázi konstrukce OBRa. Hráč tedy určí váhu pancíře jeho OBRa, která může být maximálně dvojnásobkem váhy vnitřní struktury.

5.2.8 Zabudování zbraní a munice

Jako všechny ostatní součásti OBRa i každá zbraň a munice má svoji váhu. Hráč si může vybrat libovolnou kombinaci zbraní, které nalezne v tabulce 2 na straně 32. Váha zbraní však nesmí být větší, než je zbývající váha OBRa. Autokanóny, kulomet a raketomety vyžadují pro své fungování munici. Každý zásobník munice váží jednu tunu a obsahuje různé množství dávek v závislosti na typu zbraně. Protože každá taková zbraň vyžaduje nejméně jeden zásobník, je váha zásobníku v editoru vždy odečtena od zbývající váhy ihned společně s váhou samotné zbraně.

Typ	Teplo	Síla	Min. dostřel	Krátký d.	Stř. d.	Dlouhý d.	Váha
Plamenomet	3	2	-	1	2	3	1
Velký laser	8	8	-	1-5	6-10	11-15	5
Střední laser	3	5	-	1-3	4-6	7-9	1
Malý laser	1	3	-	1	2	3	1
Vrhač částic	10	10	3	1-6	7-12	13-18	7
Autokanón/2	1	2	4	1-8	9-16	17-24	6
Autokanón/5	1	5	3	1-6	7-12	13-18	8
Autokanón/10	3	10	-	1-5	6-10	11-15	12
Autokanón/20	7	20	-	1-3	4-6	7-9	14
Kulomet	0	2	-	1	2	3	1
RDD-5	2	1/zás	6	1-7	8-14	15-21	2
RDD-10	4	1/zás	6	1-7	8-14	15-21	5
RDD-15	5	1/zás	6	1-7	8-14	15-21	7
RDD-20	6	1/zás	6	1-7	8-14	15-21	10
RKD-2	2	2/zás	-	1-3	4-6	7-9	1
RKD-4	3	2/zás	-	1-3	4-6	7-9	2
RKD-6	4	2/zás	-	1-3	4-6	7-9	3

Tabulka 2: Tabulka zbraní a vybavení.

5.2.9 Dokončení tabulky kritických zásahů

Všechny části OBRova vybavení musí být umístěny v některé části jeho těla. Každá z nich zabírá určité místo, které je reprezentováno počtem kritických pozic, na kterých je vybavení zaznamenáno. V této fázi hráč určuje, kde se které vybavení nachází. Trysky, munice a chladiče zabírají vždy jednu kritickou pozici, ovšem některé zbraně jich zabírají více. Hráč musí postupně umístit každou trysku, zbraň a zásobník na munici do některé z částí OBRa. Chladiče se neumisťují všechny, ale pouze část z nich, protože některé jsou zabudovány přímo v reaktoru. Počet chladičů, které jsou součástí reaktoru se zjistí vydělením výkonu reaktoru dvaceti pěti a zaokrouhlením výsledku dolů. Zbylé chladiče pak musí hráč rovněž umístit do některé části OBRova těla.
Důležité také je, že munice se nemusí nacházet ve stejné části OBRa jako zbraň, pro kterou je určena.

5.2.10 Vyznačení hodnot pancíře

V této fázi hráč rozdělí pancíř do celkem jedenácti oblastí na OBRově těle. Samotné rozdělení je čistě na něm, ovšem maximální množství bodů pancíře v jedné oblasti je určeno počtem bodů vnitřní struktury této oblasti. Kromě hlavy, ve které může být až 9 bodů pancíře, může být počet bodů pancíře v každé části maximálně dvojnásobek bodů vnitřní struktury. U trupu a boků navíc platí, že pancíř musí být rozdělen mezi přední a týlový pancíř, přičemž součet předního a týlního pancíře nesmí přesáhnout dvojnásobek bodů vnitřní struktury.

5.2.11 Dokončení palubní karty

V poslední části konstrukce nového OBRa už není třeba žádná interakce s uživatelem, pouze se dokončí potřebné operace a vytvoří se nový XML soubor, který obsahuje údaje o novém OBRovi.

6 Plánovaná vylepšení

Když jsem začínal s tímto projektem, měl jsem jistou představu o tom, jak by asi mohl vypadat výsledek. Věřil jsem, že se mi podaří vytvořit relativně kompletní obdobu deskové hry BattleTech. V průběhu vývoje jsem byl donucen od tohoto plánu upustit a uchýlit se k implementaci pouze zjednodušené verze hry. Důvodem byl samozřejmě především čas a pak také fakt, že jsem se s podobným projektem dosud nesetkal. V této části bych proto rád nastínil některá možná budoucí vylepšení, rozdělená do několika oblastí.

První oblastí je grafická stránka hry. Při vývoji jsem se zaměřil spíše na funkčnost grafických prvků, než na samotnou estetiku provedení, a proto je grafika velmi jednoduchá. Jejím hlavním účelem však bylo zajistit hratelnost pro hráče, tedy poskytnout mu informace o hře. Do budoucna by ovšem hra určitě měla mít lepší grafické prvky, aby působila moderněji a celkově lépe. Hlavními adepty pro toto vylepšení jsou figurky OBRů a také herní mapa.

Velká řada vylepšení je možná v samotné hře. Především se jedná o přidání fází herního kola. Kompletní verze hry totiž obsahuje navíc ještě fáze *reakce* a *fyzický útok*. Fáze reakce se odehrává bezprostředně po fázi pohyb a každý OBR může během ní natočit svůj trup o jednu hranu doleva či doprava. Na jeho budoucí směr pohybu to nemá vliv, změní se tím však jeho zorné a palebné pole. Po střelbě ze zbraní následuje fáze fyzický útok, během které může OBR napadnout jiného OBRa fyzicky. Mezi možné fyzické útoky patří například klasický úder pěstí nebo kopnutí, bodyček, úder kyjem nebo také velice zajímavý útok z výšky, kdy se útočník vznese do vzduchu pomocí zdvižných trysek a „spadne“ na svůj cíl.
Dalším možným vylepšením by mělo být zapracování kritických zásahů a odpovídajících efektů. Implementace OBRů s touto možností počítá a každý OBR, včetně těch vytvořených hráčem, má kompletní tabulku kritických pozic. Tím by bylo do hry přidáno mnoho nových možností, například zničení soupeřovy zbraně, zničení jeho chladičů či zdvižných trysek nebo dokonce zničení reaktoru a tím i celého OBRa.
Do hry by také mohlo být přidáno mnoho dalších možností a prvků, které by jí dodaly na komplexnosti. Mezi ně patří například možnost zalehnutí OBRa na zem, možnost pádu OBRa kvůli poškození či větší využití OBRova warriora, který může být zraněn nebo i zabit, což způsobí vyřezení OBRa. Velice zajímavé by bylo přidání vlivů počasí, například mlhy nebo deště, a zavedení času. Každé herní kolo by představovalo období jedné či dvou hodin a v určitý čas by začala noc, což by mělo vliv na viditelnost a především na přesnost střelby.
Herní svět BattleTechu také přímo vybízí k vytváření nejrůznějších kampaní a scénářů bitev, podle kterých lze hrát.

Úpravy a vylepšení by měly být provedeny také v samotné implementaci. Chyby v návrhu či neznalost některých zákonitostí jazyka Java během vývoje vedly k některým řešením, která nejsou optimální. Například dělení kódu v herní logice na část pro hru přes síť a část pro lokální hru by se dala vyřešit lépe. Přestože řešení těchto dvou částí se

musí do jisté míry lišit, některé funkce by šlo řešit jednotným způsobem. Současný stav je důsledkem špatného návrhu a mé snahy o to, aby to především fungovalo, s tím, že do budoucna dojde k optimalizaci kódu.
Další vylepšení je potřeba provést v implementaci komunikace při hře přes síť. V současném stavu totiž dojde k zablokování okna klienta, který čeká na reakci soupeře, kterou mu má poslat server. Toto chování však určitě není žádoucí, protože během tohoto čekání hráč nemůže dělat vlastně nic. Lepším řešením této situace by bylo použití dalších vláken, která by obstarávala komunikaci mezi serverem a klienty. Takové vlákno by tedy plnilo úlohu jakéhosi prostředníka, díky čemuž by nedocházelo k zablokování okna čekajícího klienta.
Jistá úprava by byla vhodná také u editoru OBRů. Přestože použitá metoda několika vyskakovacích oken pro zadávání potřebných údajů má své výhody, při zpětném pohledu se domnívám, že by bylo lepším řešením například zobrazení formulářů ve středovém panelu obrazovky, místo zobrazení v dialogových oknech.
Již dříve jsem zmiňoval, že hra proti UI je momentálně možná pouze prostřednictvím serveru a probíhá tedy podobně, jako hra proti jinému hráči přes síť. Rád bych však přidal možnost hry proti UI bez nutnosti připojení k serveru. Hráč by tak mohl hrát sám i bez připojení k internetu, což by mimo jiné snížilo nároky na server. Samotná implementace UI by navíc mohla být také dále vylepšena. Přestože se má jednat pouze o jednoduchou UI, bylo by dobré mít například k dispozici různě schopné varianty UI, proti kterým by hráč mohl hrát. To by znamenalo přepracování algoritmu pro volbu cíle i pro volbu zbraní, kde by UI brala v úvahu pravděpodobnost zásahu nebo aktuální stav přehřátí OBRa.

7 Závěr

V této práci jsem se snažil o vytvoření vlastní verze hry BattleTech. Přestože výsledek úplně nenaplňuje moje počáteční očekávání, vývoj této aplikace mi přinesl velké množství zkušeností jak s programováním konkrétně v jazyce Java, tak i obecně s programováním jako takovým. Seznámil jsem se s mnoha možnostmi, jak řešit nejrůznější problémy, blíže se seznámil s některými základy programování větších projektů a také jsem se naučil některým zajímavým programátorským technikám. Velmi zajímavou výzvou pro mě bylo programování umělé inteligence, kdy jsem musel převést běžné lidské uvažování do algoritmu, který by byl implementovatelný a samozřejmě účinný. S něčím podobným jsem se dosud nesetkal a můžu říct, že to byla velice zajímavá zkušenost.

Práce na tomto projektu mě opravdu bavila. Mrzí mě ovšem, že jsem na jeho vývoj neměl více času, aby se tak jeho konečná podoba více přiblížila mým původním cílům. Navíc nedostatek předchozích zkušeností s takovýmito typy projektů způsoboval občasné zdržení kvůli nutnosti přepisování kódu v důsledku zavedení nových funkcí. Celkově však pro mne práce na této aplikaci byla velmi přínosná a při zpětném pohledu můžu říct, že bych na ní rád začal pracovat i dříve než na začátku posledního ročníku bakalářského studia.

Doufám, že zkušenosti, získané při vývoji této aplikace, budu moci dále rozvíjet a využívat v dalším studiu i v praxi. Také bych rád pokračoval s vývojem a vylepšováním této aplikace, i když to pravděpodobně bude pouze pro mé osobní využití. Šlo by tedy především o další sebevzdělávání.
Také doufám, že v budoucnu dojde k nasazení této hry na plánovaný herní portál, což by mimo jiné pro mě znamenalo další inspiraci k rozšiřování aplikace a rovněž možnost získat odezvu od skutečných hráčů, což by velice pomohlo v dalším vývoji. Pokud by k nasazení mělo dojít, rád bych do vývoje případně zapojil více osob a spolupracoval s nimi v týmu, protože komplexnost BattleTechu je obrovská, s čímž souvisí i značná složitost některých herních prvků a s nimi spojené implementace. Práce v týmu na větším projektu by navíc byla další přínosnou zkušeností.

8 Reference

[1] Stránky světa BattleTech, *http://bg.battletech.com/*

[2] Red Blob Games, *http://www.redblobgames.com/grids/hexagons/*.

[3] Linear Interpolation, *http://en.wikipedia.org/wiki/Linear_interpolation*

[4] Red Blob Games - Lineární interpolace v mřížce šestiúhelníků, *http://www.redblobgames.com/grids/line-drawing.html#interpolation*.

[5] Red Blob Games - Souřadnicové systémy v mřížce šestiúhelníků, *http://www.redblobgames.com/grids/hexagons/#coordinates*.

[6] Breadth-first search algorithm, *http://en.wikipedia.org/wiki/Breadth-first_search*.

A Seznam příloh

Součástí bakalářské práce je CD, na kterém se nacházejí následující adresáře:

- */BattleTech* - obsahuje vše pro spuštění hry
- */Implementation* - obsahuje implementaci hry BattleTech v IDE Eclipse
- */Thesis* - obsahuje text bakalářské práce ve formátu PDF